Design and Analysis of Algorithms Assignment- 4

Harsha Vardhan Madasi-IIT2019108
Rahul Kumar-IIT2019109
Sumit Katiyar-IIT2019110
B.Tech. 4th semester, Department of Information Technology
Indian Institute of Information Technology, Allahabad

***Abstract*— Given an array of 2n elements in the following format a1, a2, a3, a4, ..., an, b1, b2, b3, b4, ..., bn . The task is shuffle the array to a1, b1, a2, b2, a3, b3, ..., an, bn without using extra space.**

I. INTRODUCTION

In this problem, we have to shuffle a given array in such a way that if the given array is a1, a2, a3, a4....., an, b1, b2, b3, b4, ..., bn. The task is shuffle the array to a1, b1,a2, b2, a3, b3, ..., an, bn using divide and conquer algorithm.

Divide and Conquer Algorithm: This technique can be divided into the following three parts:

- **Divide:** This involves dividing the problem into some sub problem.
- **Conquer:** Sub problem by calling recursively until sub problem solved.
- **Combine:** The Sub problem Solved so that we will get find problem solution.

This report further contains:-

1) Algorithm Design
2) Algorithm Analysis
3) Profiling
4) Conclusion

II. ALGORITHM DESIGN

Basically, we have come up with an approach Using divide and conquer algorithm

Algorithm: Divide and Conquer

In this algorithm we divide the given array into half (say arr1[] and arr2[]) and swap second half element of arr1[] with first half element of arr2[]. Recursively do this for arr1 and arr2.

For example:

- Let us consider an array be a1, a2, a3, a4, b1, b2, b3, b4.
- Now split the array into two halves i.e., a1, a2, a3, a4 ; b1, b2, b3, b4.
- Now we will swap elements which are present around the center i.e., we will swap a3, a4 with b1, b2 correspondingly.
- After swapping we will get: a1, a2, b1, b2, a3, a4, b3, b4
- Now we will recursively split the above subarrays and swap the elements around the center for each subarray.
- Finally we will get a1, b1, a2, b2 and a3, b3, a4, b4.

PSEUDO CODE:

```
//Function to shufle an array
Void shuflearray(int arr[],int start , int end)
// base condition
        if(end <start )then
            return ;

//if only two elements present in the subarray
        if(end = start + 1)then
            return ;

//Finding middle of the array to divide the array
int mid ← (start + end) /2;

//using temp in order to swap first half of second array
int temp ← mid +1;

//using first in order to swap second half of first array
int firstmid ← (start + mid)/2;

//Swapping the center elements
for(i = firstmid +1 to i <= mid )
        swap (arr[i],arr[temp++]);
        i ← i+1;

// recursively calling the function for first and second subarrays
shuflearray(arr, start, mid);
shuflearray(arr, mid + end, l);
```

//MAIN FUNCTION

```
int main()
    int n;
    input n;
    int arr[n];
```

```
//taking input
for(int i = 0 to n )
    input arr[i];
shufflearray (arr,0,n-1);
//printing the final shuffled array
for(int i = 0 to n )
    print arr[i];
return 0;
```

III. ALGORITHM ANALYSIS :

Time complexity:
In the above algorithm we get the recurrence
T(n)=2T(n/2)+O(n)
for the time complexity, which results in
T(n)=O(nlogn)
So the final computational time to shuffle the given array is O(nlogn).

Space complexity:
To be precise,as we are swapping the elements in the given array itself there is no extra space required.So the space complexity will be O(1).

IV. PROFILING :

Posteriori Analysis So after the above discussion of the time and space complexity in asymptotic analysis , we come to the profiling part.

Following given is the graphical representation of time complexity of the Algorithm approach given below.

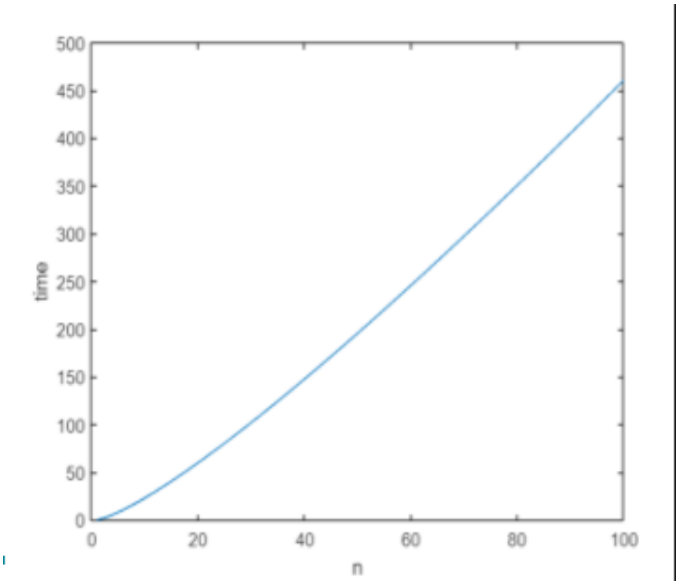

Fig. 1. Time complexity graph(O(nlogn))

V. CONCLUSION

Hence, from the above graph and analysis, we come to the conclusion that this problem can be solved using the time complexity of O(nlogn) and space complexity of O(1).

VI. REFERENCES

1) Introduction to Algorithms by Cormen,Charles,Rivest and Stein.

2) Geeksfor Geeks.